PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-39328-1660406443

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT DONE
BY

SAJITH P(922519104134)
SIVA SUBRAMANIAM B(922519104151)
RAMSANTHOSH S(922519104126)
SUDARSON K(922519104162)

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

INDEX

3. Reports from JIRA

7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    1. Feature 1
    2. Feature 2
    3. Database Schema (if Applicable)
8. **TESTING**
    1. Test Cases
    2. User Acceptance Testing
9. **RESULTS**
    1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

In simple words, personal finance entails all the financial decisions and activities that a Financeapp makes your life easierby helping you to manage your financesefficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will benotified with an email alert.

1.2 PURPOSE

A comprehensive money management strategy requires clarity and conviction for decision-making. You will need a defined goal and a clear vision for grasping the business and personal finances. That’s when an expense tracking app comes into the picture.

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate recordof your money inflow and outflow. Many people in India live on a fixed income,and they find that towardsthe end of the month theydon'thave sufficient money to meet their needs.

CHAPTER-2
LITERATURE SURVEY

2.1 EXISTING PROBLEM

1. Miriam Thomas et al., proposed an Expense Tracker System which works based on the Least Square Algorithm which is a statistical procedure to find the best fit for a data points by minimizing offsets.In this,they have proposed an application which allows the user to maintain a Digital Automated Diary.The Useris required to register on the system to get an use id and login password which they will use to keep track of their expenses.

2.Gomathy et al., proposed a system which has an Expense Tracker with few more features like Weekly Budget planner to keep track of expenses,UPI linkup to keep track ofonline transactions and an Automated message alert will be generated when the user crosses their budget limit,Wishlist,Rewards,Weekly and Monthly Analysis in the form of a piechart.

2.2 REFERENCES

1.International Journal for research in Applied Science & Engineering Technology (IJRASET) ExpenseTracker Aman Garg , Mukul Goel , Sagar Mittal, Mr, Shekhar Singh.

2.EXPENDITURE MANAGEMENT SYSTEM Dr. C.K.Gomathy, G.Nikhitha ,H.Sri Lasya , Dr. V. Geetha.

3.https://www.researchgate.net/publication/36062004

4.https://ijirt.org/master/publishedpaper/IJIRT150860_PAPER.pdf

2.3 PROBLEM STATEMENT DEFINITION

Expense Tracker isan Application which can help the user to keep track of their Expenses.Nowadays, people can do variousthings by using a mobile and so,they can also use it for Budgeting and planning their expense in the mobile instead of doing it manually.For this purpose ,an application can be developed to satisfy the needs of the customer. This application can help the user to keep track of their expenses in an organized way and to maintaina proper balance between expenditure andsavings.

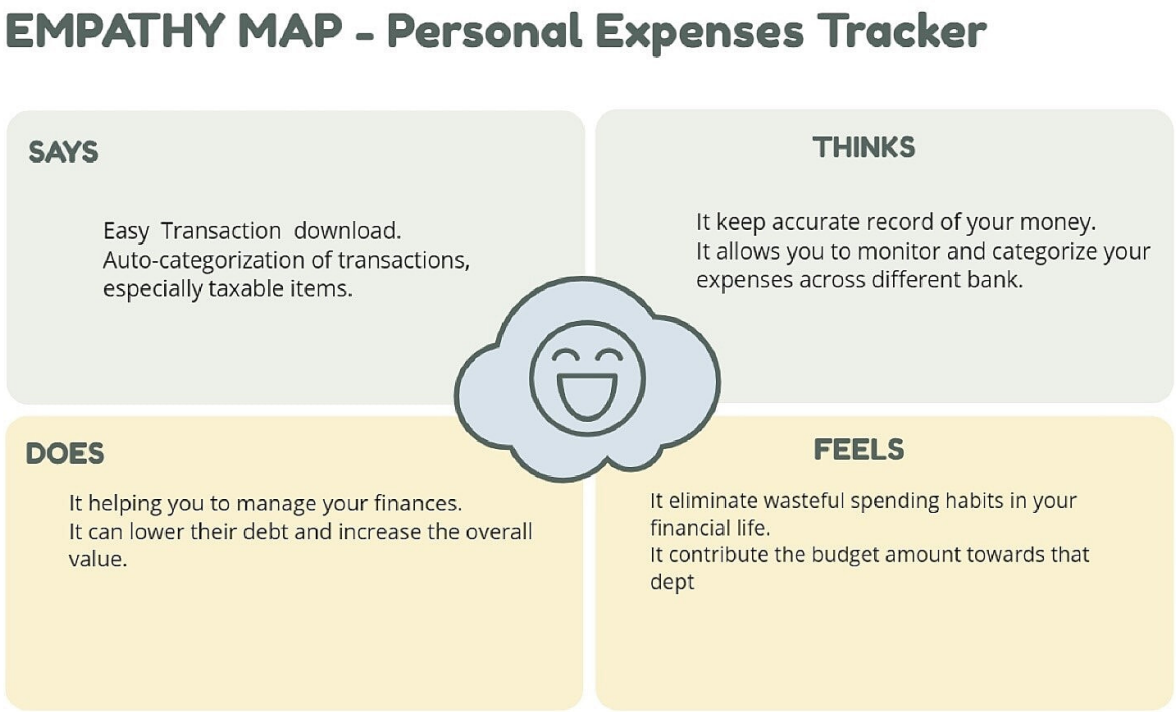
In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will benotified with an email alert.

CHAPTER-3
IDEATION & PROPOSED SYSTEM

3.1 EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’sbehaviours and attitudes. It is a useful tool to helps teams better understand their users.Creating an effective solutionrequires understanding the true problemand the person who is experiencing it. The exerciseof creating the map helpsparticipants consider thingsfrom the user’sperspective along with his or her goals and challenges.



3.2 IDEATION AND BRAINSTORMING

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

SAJITH

- Login To The Application
- Add Income
- Set Budget
- Connect This To Payment Apps

SIVASUBRAMANIAM

- category Your Expenses
- Virtualize The Expenses
- See Expense Graphically
- Edit Expenses Per Day

RAMSANTHOSH

- Generate Daily Report
- Save some amount Of Salary For Exceptional Cases
- Show separatly income and expense
- Keep The Records Backed Up

SUDARSON

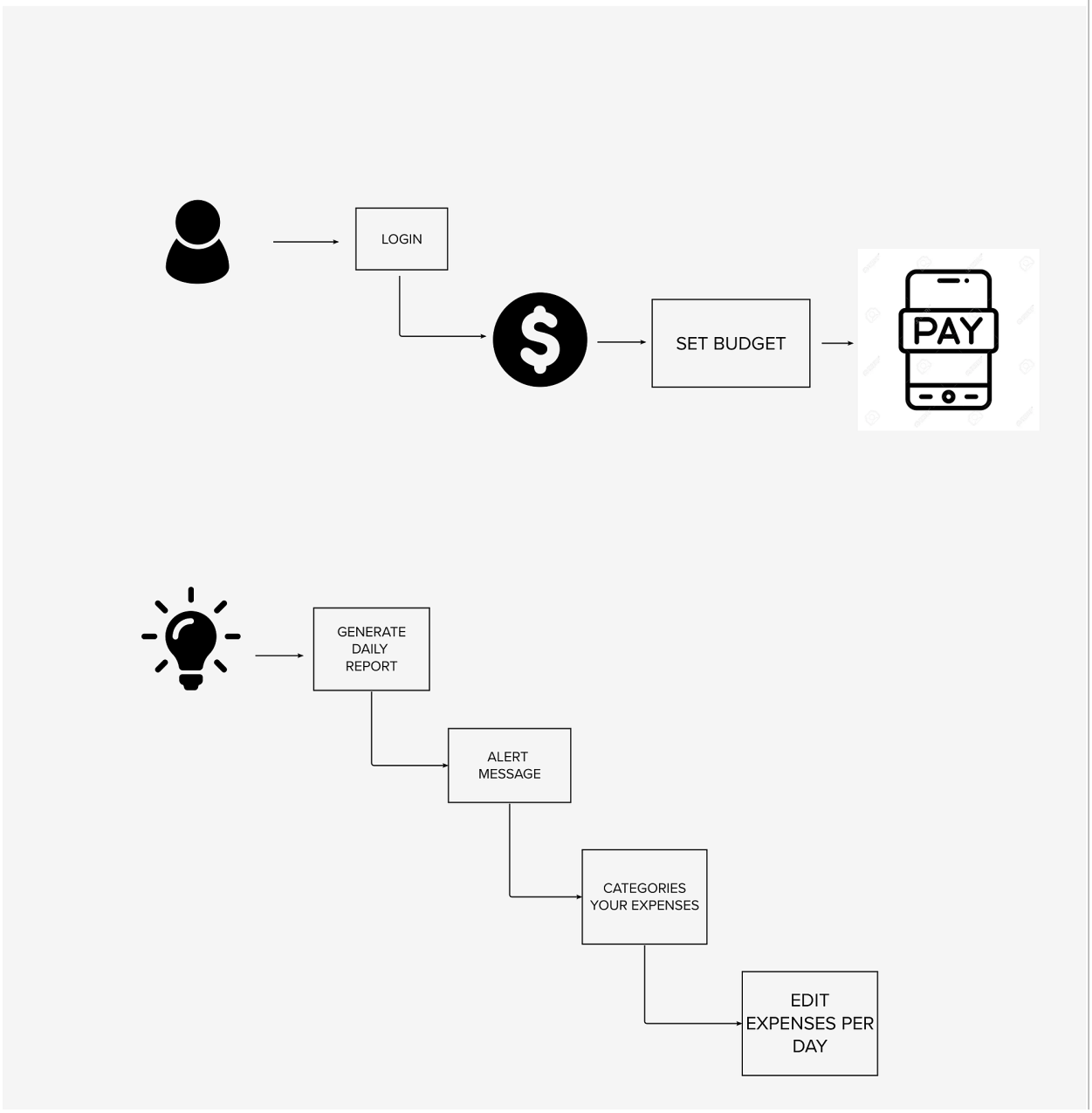
- Alert Message
- Restrict Expense The Next Day
- Try Not To Be ExtraVagent
- Analyze Your Expense At The End Of The Day

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

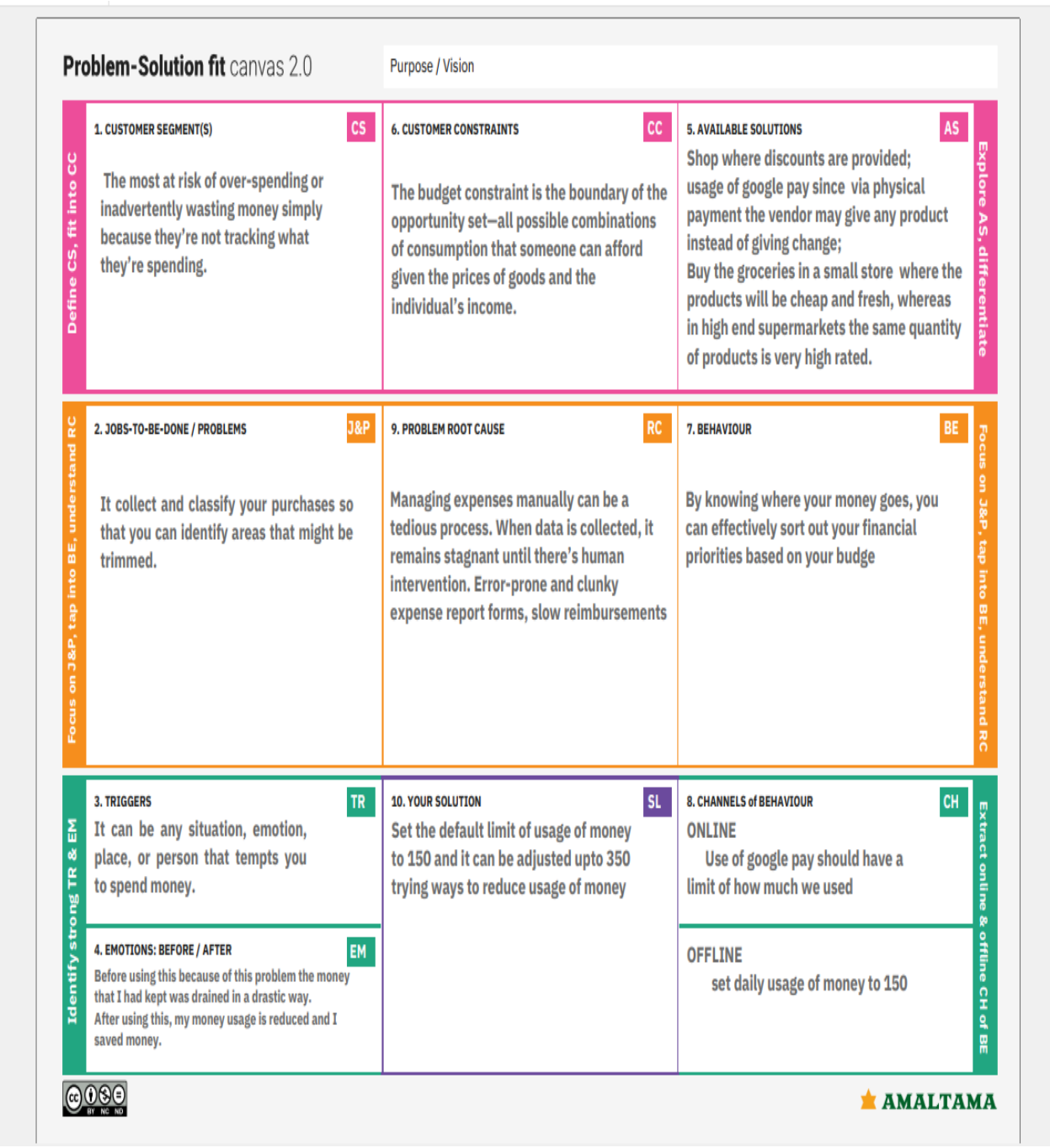
3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	ProblemStatement(Problemtobesolved)	Many organizations have their own system torecord their incomeand expenses, which they feel is the main key point of their business progress. It is good habit for a person to record daily expenses and earning but due to unawarenessand lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to doso. Due to lack of a complete tracking system, there is a 2 constant overload to rely on the daily entryof the expenditure and total estimation till the end of the month.
2.	Idea/Solutiondescription	We are building an android application named as “Expense Tracker”. As the name suggests, this project is an android app which is used to track the daily expenses of the user. It is like digital recordkeeping which keeps the recordsof expenses done by a user. The application keepsthe track of the Income and Expenses both of useron a day-to-day basis. This application takes the income of a user and manage its daily expenses sothat the user can save money. If you exceed dailyexpense allowed amount it will give you a warning, so that you don’t spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into user’s savings. The application generates report of the expenses of each end of the month. The amount saved can be used for celebrating festivals, Birthdays or Anniversary.

3.	Novelty/ Uniqueness	It will have various options to keep record (forexample Food, Travelling Fuel, Salary etc.). Automatically it will keep on sending notificationsfor our daily expenditure. In today's busy and expensive life,we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit. Hereuser can define their own categories for expense type likefood, clothing, rent and bills where they haveto enter the money that has been
		spend and likewise can add some data in extra datato indicate the expense.
4.	SocialImpact/ CustomerSatisfaction	Money is the significant sourceof stress for nearly two-thirds of Americans. Fortunately, you don't have to wait until income increases or your debts are gone to enjoy relief- the mere act of planning ahead can reap immediate benefits. Here are somesimple strategies you canuse to better track and budget your expenses.
5.	BusinessModel(RevenueModel)	A well-known personal expense tracker, Mint is also a simple tool for smaller businesses and freelancers to track where money is going. It lets you create budgets and goals within the app, andtrack your credit score. You can access all of thisdatathrough an easy-to-read dashboard, so you know your standing at any time.

6.	Scalabilityofthe Solution	Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what's to come. On the offchance that you know precisely where your sum isgoing much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will help us to keep record of our money daily. The project what we havecreated is work more proficient than the other income andexpense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain thepay and cost each month. It's a user-friendly application.

3.4 PROBLEM SOLUTION FIT



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNTIONAL REQUIREMENT

FR No.	Functional Requirement(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Financial Accounts	Account Details Verification of Details
FR-4	User Dashboard	Expense Data Data Records
FR-5	User Notifications	System Access Real time Alerting
FR-6	Security of UserData	Secured Database Data Security Algorithms

4.2 NON-FUNCTIONAL REQUIREMENTS

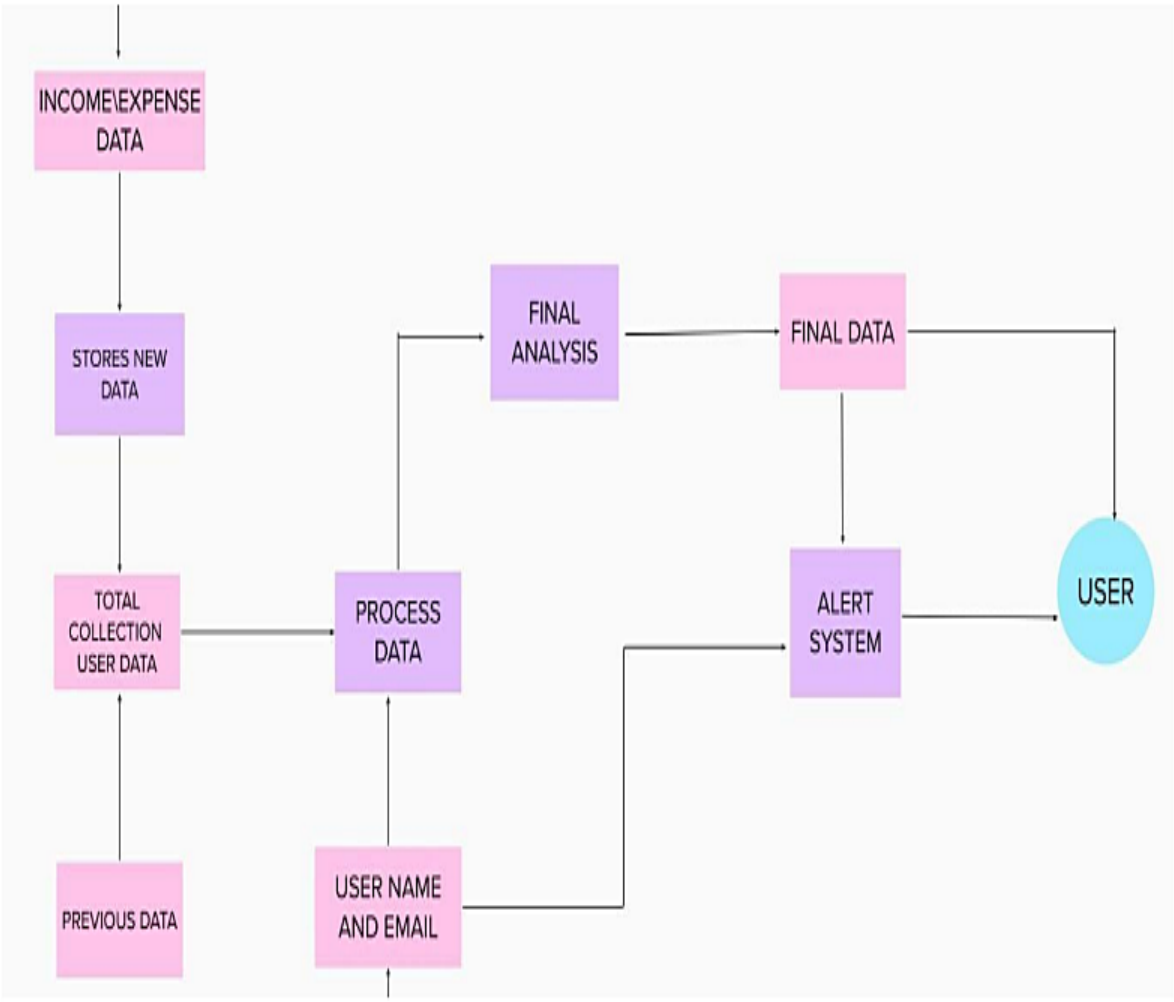
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	By using this application, the user can keep track of their expenses and can ensure that user's money is usedwisely.
NFR-2	**Security**	Maintain user personal details in a encrypted manner by using data securityalgorithms .
NFR-3	**Reliability**	It will maintain a proper tracking of day-to-dayexpenses in an efficient manner.
NFR-4	**Performance**	By enter our incoming and departing cash, and the software can help you keep and monitor it with at-most quality and security with highperformance.
NFR-5	**Availability**	Using chartsand graphs may help youmonitoryour budgeting and assets.
NFR-6	**Scalability**	Rely on your budgeting app to track,streamline, and automate all the recurrent expenses and remindyou on a timely basis.

CHAPTER 5

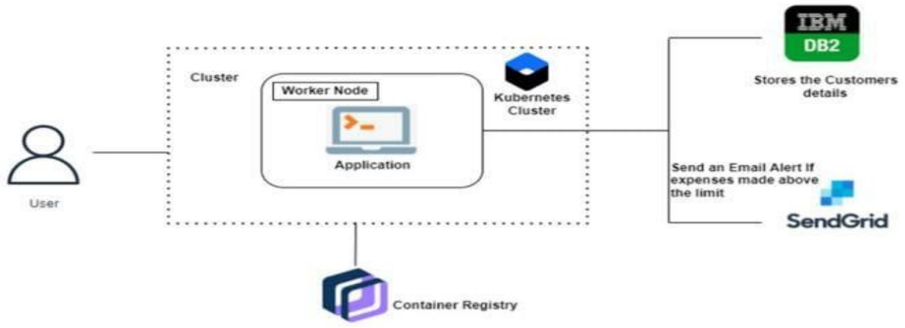
PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

Data Flow Diagrams:



5.2 SOLUTION & TECHNICAL ARTCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement(Epic)	User story number	User Story/Task	Acceptance criteria	Priority	Release
Customer (Mobileuser)	Registration	UNS-1	As a user, I can register for the application by entering my email, password,and confirming my password	I can access my account/dashboard	High	
	Login	USN-2	As a user , I can log into the application by entering email & password	I can accessthe application	High	
	Dashboard	UNS-3	As a userI can entermy income and expenditure details	I an viewmy daily expenses	High	
Customer care Executive		UNS-4	As a customer areexecutive , I can solve the log in issuesand other issues of the application	I can provide support or solution at any time24*7	Medium	
Administrator	Application	UNS-5	As an administrationI can upgrade or update the application	I can fix the bug which arisesfor the customer and users of theapplication		

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I canregister for theapplication by entering my email, password, and confirming my password.	2	High	P.Sajith S.Ramanthosh
Sprint-1		USN-2	As a user,I will receiveconfirmation email once I have registered for the application	1	High	Sivasubramaniam
Sprint-2		USN-3	As a user, I can register for the applicationthrough Facebook	2	Low	K.Sudarson P.Sajith
Sprint-1		USN-4	As a user, I can register for the applicationthrough Gmail	2	Medium	S.Ramsanthosh Sivasubramaniam
Sprint-1	Login	USN-5	As a user, I can loginto the application byentering email& password	1	High	K.Sudarson
Sprint -2	Dashboard	USN-6	As a userafter logged in,I wished to see my wallet page.	1	Low	SIvasubramaniam
Sprint-2		USN-7	As a user, I can add expense underexpensepage.	2	High	S.Ramsanthosh K.Sudarson
Sprint-3	Backend	USN-8	As a developer, I need to create backenddatabase for storing information.	1	High	P.Sajith
Sprint-3		USN-9	As a developer, automate the mailto send alert when expense reachthe limit.	1	Medium	S.Ramsanthosh

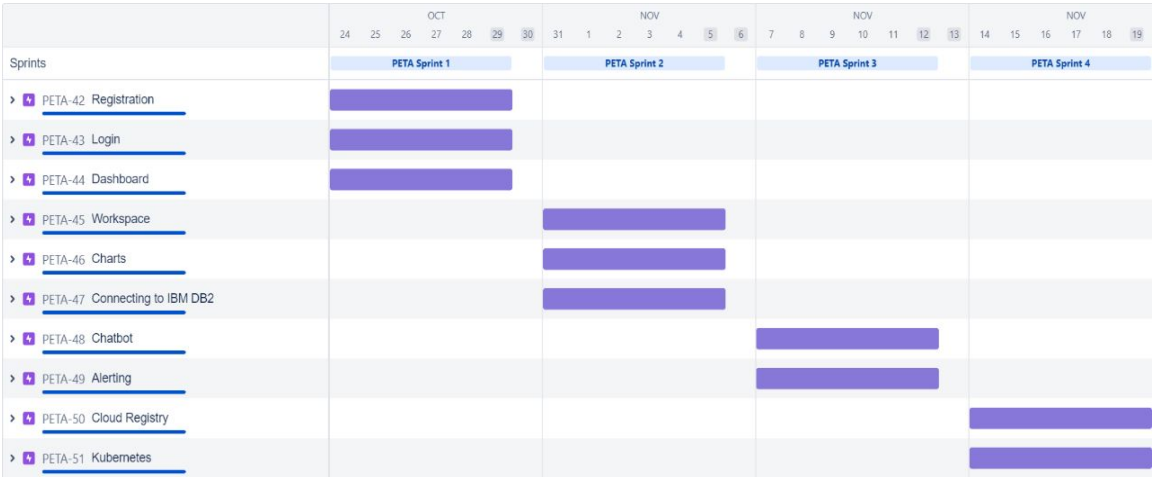
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Containerization &Testing	UNS-10	As a developer, Needto container the project inthe professional way to workevery where.	2	High	K.Sudarson S.Ramsanthosh
Sprint-4		USN-11	As a developer, test the project to check whether the project correctly work or not.	2	High	P.Sajith Sivasubramaniam

6.2 SPRINT DELIVERY SCHEDULE

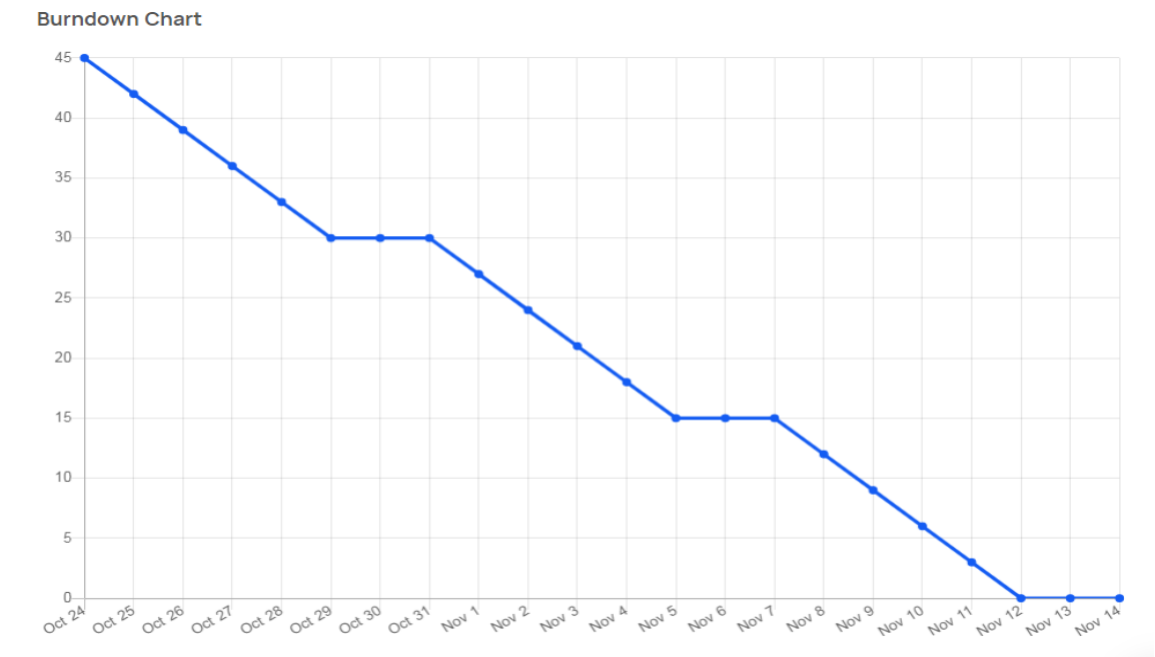
-

Sprint	Total Story points	Duration	SprintStart Date	SprintEnd Date (Planned)	Story Points Completed (as on Planned End Date)	SprintRelease Date (Actual)
Sprint-1	20	6Days	24Oct 2022	29 Oct2022	20	29 Oct2022
Sprint-2	20	6Days	31Oct 2022	05 Nov2022	20	05 Nov2022
Sprint-3	20	6Days	07Nov 2022	12 Nov2022	20	12 Nov2022
Sprint-4	20	6Days	14Nov 2022	19 Nov2022	20	19 Nov2022

6.3 REPORT FROM JIRA



BURNDOWN CHART



CHAPTER 7

CODING AND SOLUTION

app.py:

```
# -*-coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
from flask import Flask, render_template, request, redirect, session

# from flask_mysqldb import MySQL

# import MySQLdb.cursors

import re

from flask_db2 import DB2

import ibm_db

import ibm_db_dbi

from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import WSGIServer

import os

app = Flask(__name__)

app.secret_key = 'a'

# app.config['MYSQL_HOST'] = 'remotemysql.com'

# app.config['MYSQL_USER'] = 'D2DxDUPBii'

# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
```

```
# app.config['MYSQL_DB'] = 'D2DxDUPBii



"""
dsn_hostname = "3883e7e4-18f5-4afe-be8c□
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
dsn_protocol = "tcpip"
dsn = (
 "DRIVER={0};"
 "DATABASE={1};"
 "HOSTNAME={2};"
 "PORT={3};"
 "PROTOCOL={4};"
 "UID={5};"
 "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid,
dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = '3883e7e4-18f5-4afe-be8c□
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
```

```
try:
 mysql = DB2(app)
 conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpip;\
id=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'

 ibm_db_conn = ibm_db.connect(conn_str,'','')

 print("Database connected without any error !!")

except:

 print("IBM DB Connection error : " + DB2.conn_errormsg())

# app.config['']

# mysql = MySQL(app)

#HOME--PAGE

@app.route("/home")

def home():

 return render_template("homepage.html")

@app.route("/")

def add():

 return render_template("home.html")

#SIGN--UP--OR--REGISTER

@app.route("/signup")

def signup():

 return render_template("signup.html")

@app.route('/register', methods =['GET', 'POST'])

def register():

 msg = ''
```

```
print("Break point1")

if request.method == 'POST' :

username = request.form['username']

email = request.form['email']

password = request.form['password']

print("Break point2" + "name: " + username + "------" + email + "------" + password)

try:

print("Break point3")

connectionID = ibm_db_dbi.connect(conn_str, '', '')

cursor = connectionID.cursor()

print("Break point4")

except:

print("No connection Established")

# cursor = mysql.connection.cursor()

# with app.app_context():

# print("Break point3")

# cursor = ibm_db_conn.cursor()

# print("Break point4")

print("Break point5")

sql = "SELECT * FROM register WHERE username = ?"

stmt = ibm_db.prepare(ibm_db_conn, sql)

ibm_db.bind_param(stmt, 1, username)

ibm_db.execute(stmt)

result = ibm_db.execute(stmt)
```

```
print(result)

account = ibm_db.fetch_row(stmt)

print(account)



param = "SELECT * FROM register WHERE username = '' + "\'" + username + "\'"
res = ibm_db.exec_immediate(ibm_db_conn, param)
print("---- ")
dictionary = ibm_db.fetch_assoc(res)
while dictionary != False:
print("The ID is : ", dictionary["USERNAME"])
dictionary = ibm_db.fetch_assoc(res)
# dictionary = ibm_db.fetch_assoc(result)
# cursor.execute(stmt)
# account = cursor.fetchone()
# print(account)
# while ibm_db.fetch_row(result) != False:
# # account = ibm_db.result(stmt)
# print(ibm_db.result(result, "username"))
# print(dictionary["username"])
print("break point 6")
if account:
msg = 'Username already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
msg = 'name must contain only characters and numbers !'
else:
sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
ibm_db.bind_param(stmt2, 1, username)
ibm_db.bind_param(stmt2, 2, email)
```

```
 ibm_db.bind_param(stmt2, 3, password)

ibm_db.execute(stmt2)

 # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)',

(username, email,password))

 # mysql.connection.commit()

 msg = 'You have successfully registered !'

 return render_template('signup.html', msg = msg)

#LOGIN--PAGE

@app.route("/signin")

def signin():

 return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])

def login():

 global userid

 msg = ''

 if request.method == 'POST' :

 username = request.form['username']

 password = request.form['password']

 # cursor = mysql.connection.cursor()

 # cursor.execute('SELECT * FROM register WHERE username = % s AND password =

% s', (username, password ),)

 # account = cursor.fetchone()

 # print (account)

 sql = "SELECT * FROM register WHERE username = ? and password = ?"
```

```
 stmt = ibm_db.prepare(ibm_db_conn, sql)
 ibm_db.bind_param(stmt, 1, username)


ibm_db.bind_param(stmt, 2, password)
 result = ibm_db.execute(stmt)
 print(result)
 account = ibm_db.fetch_row(stmt)
 print(account)
 param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'" + "
and password = " + "\'" + password + "\'"
 res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)
 # sendmail("hello sakthi","sivasakthisairam@gmail.com")
 if account:
 session['loggedin'] = True
 session['id'] = dictionary["ID"]
 userid = dictionary["ID"]
 session['username'] = dictionary["USERNAME"]
 session['email'] = dictionary["EMAIL"]
 return redirect('/home')
 else:
 msg = 'Incorrect username / password !'
 return render_template('login.html', msg = msg)
#ADDING----DATA
```

```
@app.route("/add")
def adding():
 return render_template('add.html')
@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
 date = request.form['date']
 expensename = request.form['expensename']
 amount = request.form['amount']
 paymode = request.form['paymode']
 category = request.form['category']
 print(date)
 p1 = date[0:10]
 p2 = date[11:13]
 p3 = date[14:]
 p4 = p1 + "-" + p2 + "." + p3 + ".00"
 print(p4)
 # cursor = mysql.connection.cursor()
 # cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s, %
s)', (session['id'] ,date, expensename, amount, paymode, category))
 # mysql.connection.commit()
 # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)
 sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category)
VALUES (?, ?, ?, ?, ?, ?)"
 stmt = ibm_db.prepare(ibm_db_conn, sql)
```

```
 ibm_db.bind_param(stmt, 1, session['id'])




ibm_db.bind_param(stmt, 2, p4)
 ibm_db.bind_param(stmt, 3, expensename)
 ibm_db.bind_param(stmt, 4, amount)
 ibm_db.bind_param(stmt, 5, paymode)
 ibm_db.bind_param(stmt, 6, category)
 ibm_db.execute(stmt)
 print("Expenses added")
 # email part
 param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
 res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)
 expense = []
 while dictionary != False:
 temp = []
 temp.append(dictionary["ID"])
 temp.append(dictionary["USERID"])
 temp.append(dictionary["DATE"])
 temp.append(dictionary["EXPENSENAME"])
 temp.append(dictionary["AMOUNT"])
 temp.append(dictionary["PAYMODE"])
 temp.append(dictionary["CATEGORY"])
 expense.append(temp)
 print(temp)
 dictionary = ibm_db.fetch_assoc(res)
 total=0
 for x in expense:
```

```
 total += x[4]
param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + "
ORDER BY id DESC LIMIT 1"
 res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)
 row = []
 s = 0
 while dictionary != False:
 temp = []
 temp.append(dictionary["LIMITSS"])
 row.append(temp)
 dictionary = ibm_db.fetch_assoc(res)
 s = temp[0]
 if total > int(s):
 msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs.
" + s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
 sendmail(msg,session['email'])
 return redirect("/display")
#DISPLAY---graph
@app.route("/display")
def display():
 print(session["username"],session['id'])
 # cursor = mysql.connection.cursor()
 # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER
```

```
BY `expenses`.`date` DESC',(str(session['id'])))

 # expense = cursor.fetchall()

 param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER

BY date DESC"

res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)

 expense = []

 while dictionary != False:

 temp = []

 temp.append(dictionary["ID"])

 temp.append(dictionary["USERID"])

 temp.append(dictionary["DATE"])

 temp.append(dictionary["EXPENSENAME"])

 temp.append(dictionary["AMOUNT"])

 temp.append(dictionary["PAYMODE"])

 temp.append(dictionary["CATEGORY"])

 expense.append(temp)

 print(temp)

 dictionary = ibm_db.fetch_assoc(res)

 return render_template('display.html' ,expense = expense)

#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])

def delete(id):

 # cursor = mysql.connection.cursor()

 # cursor.execute('DELETE FROM expenses WHERE id = {0}'.format(id))
```

```
 # mysql.connection.commit()

 param = "DELETE FROM expenses WHERE id = " + id

 res = ibm_db.exec_immediate(ibm_db_conn, param)

 print('deleted successfully')

return redirect("/display")
#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])

def edit(id):

 # cursor = mysql.connection.cursor()

 # cursor.execute('SELECT * FROM expenses WHERE id = %s', (id,))

 # row = cursor.fetchall()

 param = "SELECT * FROM expenses WHERE id = " + id

 res = ibm_db.exec_immediate(ibm_db_conn, param)

 dictionary = ibm_db.fetch_assoc(res)

 row = []

 while dictionary != False:

 temp = []

 temp.append(dictionary["ID"])

 temp.append(dictionary["USERID"])

 temp.append(dictionary["DATE"])

 temp.append(dictionary["EXPENSENAME"])

 temp.append(dictionary["AMOUNT"])

 temp.append(dictionary["PAYMODE"])

 temp.append(dictionary["CATEGORY"])

 row.append(temp)
```

```
 print(temp)

 dictionary = ibm_db.fetch_assoc(res)

 print(row[0])

 return render_template('edit.html', expenses = row[0])

@app.route('/update/<id>', methods = ['POST'])
def update(id):
 if request.method == 'POST' :
 date = request.form['date']
 expensename = request.form['expensename']
 amount = request.form['amount']
 paymode = request.form['paymode']
 category = request.form['category']
 # cursor = mysql.connection.cursor()
 # cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s ,
`amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date,
expensename, amount, str(paymode), str(category),id))
 # mysql.connection.commit()
 p1 = date[0:10]
 p2 = date[11:13]
 p3 = date[14:]
 p4 = p1 + "-" + p2 + "." + p3 + ".00"
 sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?,
category = ? WHERE id = ?"
 stmt = ibm_db.prepare(ibm_db_conn, sql)
 ibm_db.bind_param(stmt, 1, p4)
 ibm_db.bind_param(stmt, 2, expensename)
 ibm_db.bind_param(stmt, 3, amount)
 ibm_db.bind_param(stmt, 4, paymode)
 ibm_db.bind_param(stmt, 5, category)
 ibm_db.bind_param(stmt, 6, id)
 ibm_db.execute(stmt)
 print('successfully updated')
```

```
 return redirect("/display")

#limit
@app.route("/limit" )
def limit():
 return redirect('/limitn')
@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
 if request.method == "POST":
 number= request.form['number']
 # cursor = mysql.connection.cursor()
 # cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'],
number))
 # mysql.connection.commit()
 sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
 stmt = ibm_db.prepare(ibm_db_conn, sql)
 ibm_db.bind_param(stmt, 1, session['id'])
 ibm_db.bind_param(stmt, 2, number)
 ibm_db.execute(stmt)
 return redirect('/limitn')
@app.route("/limitn")
def limitn():

# cursor = mysql.connection.cursor()
 # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
```

```
 # x= cursor.fetchone()
 # s = x[0]
 param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + "
ORDER BY id DESC LIMIT 1"
 res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)
 row = []
 s = " /-"
 while dictionary != False:
 temp = []
 temp.append(dictionary["LIMITSS"])
 row.append(temp)
 dictionary = ibm_db.fetch_assoc(res)
 s = temp[0]
 return render_template("limit.html" , y= s)
#REPORT
@app.route("/today")
def today():
 # cursor = mysql.connection.cursor()
 # cursor.execute('SELECT TIME(date) , amount FROM expenses WHERE userid =
%s AND DATE(date) = DATE(NOW()) ',(str(session['id'])))
 # texpense = cursor.fetchall()
 # print(texpense)
 param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " +
```

```
str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
 res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
 dictionary1 = ibm_db.fetch_assoc(res1)
 texpense = []
while dictionary1 != False:
 temp = []
 temp.append(dictionary1["TN"])
 temp.append(dictionary1["AMOUNT"])
 texpense.append(temp)
 print(temp)
 dictionary1 = ibm_db.fetch_assoc(res1)
 # cursor = mysql.connection.cursor()
 # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
 # expense = cursor.fetchall()
 param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
DATE(date) = DATE(current timestamp) ORDER BY date DESC"
 res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)
 expense = []
 while dictionary != False:
 temp = []
 temp.append(dictionary["ID"])
 temp.append(dictionary["USERID"])
 temp.append(dictionary["DATE"])
 temp.append(dictionary["EXPENSENAME"])
 temp.append(dictionary["AMOUNT"])
 temp.append(dictionary["PAYMODE"])
 temp.append(dictionary["CATEGORY"])
 expense.append(temp)
 print(temp)
 dictionary = ibm_db.fetch_assoc(res)
```

```
total=0

t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0
for x in expense:
total += x[4]
if x[6] == "food":
t_food += x[4]
elif x[6] == "entertainment":
t_entertainment += x[4]
elif x[6] == "business":
t_business += x[4]
elif x[6] == "rent":
t_rent += x[4]
elif x[6] == "EMI":
t_EMI += x[4]
elif x[6] == "other":
t_other += x[4]
print(total)
print(t_food)
print(t_entertainment)
```

```
 print(t_business)
 print(t_rent)
print(t_EMI)
 print(t_other)
 return render_template("today.html", texpense = texpense, expense = expense, total =
total ,
 t_food = t_food,t_entertainment = t_entertainment,
 t_business = t_business, t_rent = t_rent,
 t_EMI = t_EMI, t_other = t_other )
@app.route("/month")
def month():
 # cursor = mysql.connection.cursor()
 # cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE
userid= %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER
BY DATE(date) ',(str(session['id'])))
 # texpense = cursor.fetchall()
 # print(texpense)
 param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE
userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND
YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
 res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
 dictionary1 = ibm_db.fetch_assoc(res1)
 texpense = []
 while dictionary1 != False:
```

```
 temp = []
 temp.append(dictionary1["DT"])
 temp.append(dictionary1["TOT"])
 texpense.append(temp)
 print(temp)
 dictionary1 = ibm_db.fetch_assoc(res1)
# cursor = mysql.connection.cursor()
 # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
 # expense = cursor.fetchall()
 param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
 res = ibm_db.exec_immediate(ibm_db_conn, param)
 dictionary = ibm_db.fetch_assoc(res)
 expense = []
 while dictionary != False:
 temp = []
 temp.append(dictionary["ID"])
 temp.append(dictionary["USERID"])
 temp.append(dictionary["DATE"])
 temp.append(dictionary["EXPENSENAME"])
 temp.append(dictionary["AMOUNT"])
 temp.append(dictionary["PAYMODE"])
 temp.append(dictionary["CATEGORY"])
 expense.append(temp)
 print(temp)
 dictionary = ibm_db.fetch_assoc(res)
 total=0
 t_food=0
```

```
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0
for x in expense:
total += x[4]
if x[6] == "food":
t_food += x[4]
elif x[6] == "entertainment":
t_entertainment += x[4]
elif x[6] == "business":
t_business += x[4]
elif x[6] == "rent":
t_rent += x[4]
elif x[6] == "EMI":
t_EMI += x[4]
elif x[6] == "other":
t_other += x[4]
print(total)
print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
```

```
 print(t_other)

 return render_template("today.html", texpense = texpense, expense = expense,

total = total ,



t_food = t_food,t_entertainment = t_entertainment,
 t_business = t_business, t_rent = t_rent,
 t_EMI = t_EMI, t_other = t_other )
@app.route("/year")
def year():
 # cursor = mysql.connection.cursor()
 # cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE
userid= %s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY
MONTH(date) ',(str(session['id'])))
 # texpense = cursor.fetchall()
 # print(texpense)
 param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses
WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp)
GROUP BY MONTH(date) ORDER BY MONTH(date)"
 res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
 dictionary1 = ibm_db.fetch_assoc(res1)
 texpense = []
 while dictionary1 != False:
 temp = []
 temp.append(dictionary1["MN"])
 temp.append(dictionary1["TOT"])
 texpense.append(temp)
 print(temp)
 dictionary1 = ibm_db.fetch_assoc(res1)
 # cursor = mysql.connection.cursor()
 # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
```

```
# expense = cursor.fetchall()

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
temp = []
temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])
temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])
temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])
temp.append(dictionary["CATEGORY"])
expense.append(temp)
print(temp)
dictionary = ibm_db.fetch_assoc(res)
total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
```

```
t_other=0
for x in expense:
total += x[4]
if x[6] == "food":
t_food += x[4]
elif x[6] == "entertainment":

t_entertainment += x[4]
elif x[6] == "business":
t_business += x[4]
elif x[6] == "rent":
t_rent += x[4]
elif x[6] == "EMI":
t_EMI += x[4]
elif x[6] == "other":
t_other += x[4]
print(total)
print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)
return render_template("today.html", texpense = texpense, expense = expense, total =
```

```
total ,
 t_food = t_food,t_entertainment = t_entertainment,
 t_business = t_business, t_rent = t_rent,
 t_EMI = t_EMI, t_other = t_other )
#log-out
@app.route('/logout')


def logout():
 session.pop('loggedin', None)
 session.pop('id', None)
 session.pop('username', None)
 session.pop('email', None)
 return render_template('home.html')
port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
 app.secret_key = os.urandom(12)
 app.run(debug=True, host='0.0.0.0', port=port)
```

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: sakthi-flask-node-deployment
spec:
 replicas: 1
 selector:
 matchLabels:
 app: flasknode
 template:
 metadata:
 labels:
```

```
 app: flasknode
 spec:
 containers:
 - name: flasknode
 image: icr.io/sakthi_expense_tracker2/flask-template2
 imagePullPolicy: Always
ports:
 - containerPort: 5000
```

flask-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
 name: flask-app-service
spec:
 selector:
 app: flask-app
 ports:
 - name: http
 protocol: TCP
 port: 80
 targetPort: 5000
 type: LoadBalancer
```

manifest.yml:

```
applications:
- name: Python Flask App IBCMR 2022-10-19
 random-route: true
 memory: 512M
 disk_quota: 1.5G
```

sendemail.py:

```
import smtplib
import sendgrid as sg
import os
from sendgrid.helpers.mail import Mail, Email, To, Content
```

```
SUBJECT = "expense tracker"

s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):

 print("sorry we cant process your candidature")


s = smtplib.SMTP('smtp.gmail.com', 587)

 s.starttls()

 # s.login("il.tproduct8080@gmail.com", "oms@1Ram")

 s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")

 message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

 # s.sendmail("il.tproduct8080@gmail.com", email, message)

 s.sendmail("il.tproduct8080@gmail.com", email, message)

 s.quit()

def sendgridmail(user,TEXT):

 # from_email = Email("shridhartp24@gmail.com")

 from_email = Email("tproduct8080@gmail.com")

 to_email = To(user)

 subject = "Sending with SendGrid is Fun"

 content = Content("text/plain",TEXT)

 mail = Mail(from_email, to_email, subject, content)

 # Get a JSON-ready representation of the Mail object

 mail_json = mail.get()

 # Send an HTTP POST request to /mail/send

 response = sg.client.mail.send.post(request_body=mail_json)
```

```
print(response.status_code)
print(response.headers)
```

Database Schema

Tables :

1.Admin:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,username VARCHAR(32) NOT NULL, email VARCHAR(32) NOT NULL,password VARCHAR(32) NOT NULL

2.Expense:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
userid INT NOT NULL, date TIMESTAMP(12) NOT NULL,expensename VARCHAR(32) NOT NULL, amount VARCHAR(32) NOT NULL,
paymode VARCHAR(32) NOT NULL,
category VARCHAR(32) NOT NULL

3.LIMIT

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,userid VARCHAR(32) NOT NULL, limit VARCHAR(32) NOT NULL

CHAPTER 8

TESTING

8.1 TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1. Go to website 2. Enter Valid username and password	Username: Kavi password: 123456	Login/Signup popup should display	Working as expected	Pass	-		Kavinaya
Loginpage_TC_002	Functional	Home Page	Verify that the error message is displayed when the user enters the wrong credentials	1. Go to website 2. Enter Invalid username and password	Username: XXXX Password: 12345	Error message should displayed	Working as expected	Pass	-		Afra
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1.Go to website 2.Enter valid credentials 3.Click Login	Username: Kavi password: 123456	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	Pass	-		Abdul Waseem
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	User should navigate to user account homepage	Working as expected	Pass	-		Jayasri
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	-		Afra
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	-		Kavinaya
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	-		Abdul Waseem
AddExpensePage_TC _OO6	Functional	Add Expense page	Verify whether user is able to add expense or not	1. Add date, expense name and other details 2.Check if the expense gets added	add rent = 6000	Application adds expenses	Working as expected	Pass	-		Jayasri

8.2 USER ACCEPTANCE TESTING

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the **personal expense tracker** project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	6	4	2	5	17
Duplicate	2	0	3	0	5
External	5	2	0	1	8
Fixed	12	2	4	20	31
Not Reproduced	0	0	1	0	1
Skipped	0	1	1	1	2
Won't Fix	1	3	2	1	8
Totals	26	12	13	25	76

3.Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	1	5
Client Application	50	5	5	40
Security	2	0	0	2

Outsource Shipping	3	0	1	2
Exception Reporting	9	0	0	9
Final Report Output	5	0	0	5
Version Control	1	0	0	1

CHAPTER 9
RESULTS

9.1 PERFORMANCE METRICS

1. Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

2. Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.

3. Organizing Taxes: Import your documents to the expense tracking app, and it will streamline your income and expenses under the appropriate tax categories.

4. Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the trackingapp sends reminders for payments and automatically matches the payments with invoices.

5. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,

6. Ecommerce integration: Integrateyour expense trackingapp wit h your eCommerce store and track your sales through payments received via multiple payment methods.

7. Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.

8. Access control: Increase your team productivity by providing access control to particular users through custom permissions.

9. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project.

10. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders.

11. In-depth insights and analytics: Provides in-built tools to generate reports with easy-to- understand visuals and graphics to gain insights about the performance of yourbusiness.

12. Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

CHAPTER 10

ADVANTAGES & DISADVANTAGES

1. Achieve your business goals with a tailored mobile app that perfectly fits your business.

2. Scale-up at the pace your business is growing.

3. Deliver an **outstanding** customer experience through additional control over the app.

4. Control the **security** of your business and customer data

5. Open **direct marketing channels** with no extra costs with methods such aspush notifications.

6. Boost the productivity of all the processes within theorganization.

7. Increase **efficiency** and **customer satisfaction** with an app aligned to their needs.

8. Seamlessly integrate with existing infrastructure.

9. Ability to provide **valuable insights**.

10. Optimize sales processes to generate **more revenue** through enhanced data collection.

CHAPTER 11

CONCLUSION

From this project, we are able to manage and keep tracking the daily expenses as well as income. While making this project, we gained a lot of experience of working as a team. We discovered various predicted and unpredicted problems and we enjoyed a lot solving them as a team. We adopted things like video tutorials, text tutorials, internet and learning materials to make our project complete.

CHAPTER 12

FUTURE

The project assists well to record the income and expenses in general. However, this project has some limitations:

1. The application is unable to maintain the backup of data once it is uninstalled.

2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system:

3. Multiple language interface.

4. Provide backup and recovery of data.

5. Provide better user interface for user.

6. Mobile apps advantage.

CHAPTER 13
APPENDIX

SOURCE CODE GITHUB LINK:https://github.com/IBM-EPBL/IBM-Project-39328-1660406443

PROJECT DEMOLINK:
https://drive.google.com/file/d/1WeLP95EaJGgTIKZmrrUCmnFAOA5u10Ld/view?usp=share_link